

```

1 // fo1/5/shapes/Line.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 /**
7  * A Line has a length and a paintChar used to paint
8  * itself on a Screen.
9  *
10 * Subclasses of this abstract class specify the direction
11 * of the line.
12 *
13 * @version 5
14 */
15
16 public abstract class Line
17 {
18     protected int length; // length in (character) pixels.
19     protected char paintChar; // character used for painting.
20
21     /**
22      * Construct a Line.
23      *
24      * @param length length in (character) pixels.
25      * @param paintChar character used for painting this Line.
26      */
27     protected Line( int length, char paintChar )
28     {
29         this.length = length;
30         this.paintChar = paintChar;
31     }
32
33     /**
34      * Get the length of this line.
35      *
36      * @return the length in (character) pixels.
37      */
38     public int getLength()
39     {
40         return length;
41     }
42
43     /**
44      * Set the length of this line.
45      *
46      * @param length the new length in (character) pixels.
47      */
48     public void setLength( int length )
49     {
50         this.length = length;
51     }
52
53     /**
54      */
55
56

```

```

57     * Get the paintChar of this Line.
58     *
59     * @return the paintChar.
60     */
61
62     public char getPaintChar()
63     {
64         return paintChar;
65     }
66
67     /**
68      * Set the paintChar of this Line.
69      *
70      * @param paintChar the new paintChar.
71      */
72
73     public void setPaintChar( char paintChar )
74     {
75         this.paintChar = paintChar;
76     }
77
78     /**
79      * Paint this Line on Screen s at position (x,y).
80      *
81      * @param s the Screen on which this Line is to be painted.
82      * @param x the x position for the line.
83      * @param y the y position for the line.
84      */
85
86     public abstract void paintOn( Screen s, int x, int y );
87
88     /**
89      * Paint this Line on Screen s at position (0,0).
90      *
91      * @param s the Screen on which this Line is to be painted.
92      */
93
94     public void paintOn( Screen s )
95     {
96         paintOn( s, 0, 0 );
97     }
98

```

```

1 // fo1/5/shapes/HLine.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 /**
7  * An HLine is a horizontal line.
8  */
9
10 public class HLine extends Line
11 {
12     /**
13      * Construct an HLine having a paintChar and a length.
14      *
15      * @param length length in (character) pixels.
16      * @param paintChar character used for painting this line.
17      */
18
19     public HLine( int length, char paintChar )
20     {
21         super( length, paintChar );
22     }
23
24     /**
25      * Paint this Line on Screen s at position (x,y).
26      *
27      * @param screen the Screen on which this Line is to be painted.
28      * @param x       the x position for the line.
29      * @param y       the y position for the line.
30      */
31
32     public void paintOn( Screen screen, int x, int y )
33     {
34         for ( int i = 0; i < length; i++ )
35             screen.paintAt( paintChar, x+i, y );
36     }
37
38     /**
39      * Unit test for class HLine.
40      */
41
42     public static void main( String[] args )
43     {
44         Terminal terminal = new Terminal();
45
46         terminal.println( "Self documenting unit test of HLine." );
47         terminal.println( "The two Screens that follow should match." );
48         terminal.println();
49         terminal.println( "Hard coded picture:" );
50         terminal.println( "+++++++" );
51         terminal.println( "+++++++" );
52         terminal.println( "+++++++" );
53         terminal.println( "+++++++" );
54         terminal.println( "+++++++" );
55         terminal.println( "+++++++" );
56         terminal.println( "+++++++" );

```

```

57         terminal.println( "+" );
58         terminal.println( "+++++++" );
59         terminal.println();
60
61         terminal.println( "Picture drawn using HLine methods:" );
62         Screen screen = new Screen( 20, 6 );
63
64         Line hline = new HLine( 10, 'x' );
65         hline.paintOn( screen );
66
67         hline.setLength( 5 );
68         hline.paintOn( screen, 0, 1 );
69
70         hline.setPaintChar( '*' );
71         hline.paintOn( screen, 3, 3 );
72
73         hline.setLength( 1 );
74         hline.setPaintChar( '1' );
75         hline.paintOn( screen, 4, 4 );
76
77         screen.draw( terminal );
78
79     }
80 }

```

```

1 // fo1/5/shapes/VLine.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 /**
7  * A VLine is a vertical line.
8  */
9
10 public class VLine extends Line
11 {
12     /**
13      * Construct a VLine having a paintChar and a length.
14      *
15      * @param length length in (character) pixels.
16      * @param paintChar character used for painting this line.
17      */
18
19     public VLine( int length, char paintChar )
20     {
21         super( length, paintChar );
22     }
23
24     /**
25      * Paint this Line on Screen s at position (x,y).
26      *
27      * @param screen the Screen on which this Line is to be painted.
28      * @param x       the x position for the line.
29      * @param y       the y position for the line.
30      */
31
32     public void paintOn( Screen screen, int x, int y )
33     {
34         for ( int i = 0; i < length; i++ )
35             screen.paintAt( paintChar, x, y+i );
36     }
37
38     /**
39      * Unit test for class VLine.
40      */
41
42     public static void main( String[] argv )
43     {
44         Terminal terminal = new Terminal();
45
46         terminal.println( "Self documenting unit test of VLine." );
47         terminal.println( "The two Screens that follow should match." );
48         terminal.println();
49         terminal.println( "Hard coded picture:" );
50         terminal.println( "+++++++" );
51         terminal.println( "+x     +");
52         terminal.println( "+xx    +");
53         terminal.println( "+xx   +");
54         terminal.println( "+xx  +");
55         terminal.println( "+xx * +");
56         terminal.println( "+xx *1 +");
57         terminal.println( "+x  * +");

```

```

57         terminal.println( "+x  * +");
58         terminal.println( "+   * +");
59         terminal.println( "+   +");
60         terminal.println( "+++++++" );
61         terminal.println();
62
63         terminal.println( "Picture drawn using VLine methods:" );
64         Screen screen = new Screen( 7, 9 );
65
66         VLine vline = new VLine( 7, 'x' );
67         vline.paintOn( screen );
68
69         vline.setLength(5);
70         vline.paintOn( screen, 1, 0 );
71
72         vline.setPaintChar( '*' );
73         vline.paintOn( screen, 3, 3 );
74
75         vline.setLength(1);
76         vline.setPaintChar( '1' );
77         vline.paintOn( screen, 4, 4 );
78
79         screen.draw( terminal );
80
81     }
82 }

```

```

1 // fo1/5/shapes/ShapeOnScreen.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5
6 // This file is used in one of the Chapter 5 exercises on shapes.
7
8 /**
9  * A ShapeOnScreen models a Shape to be painted at
10 * a given position on a Screen.
11 *
12 * @see Shape
13 * @see Screen
14 * @version 5
15 */
16
17
18 public class ShapeOnScreen
19 {
20     private Shape shape;
21     private int x;
22     private int y;
23
24     /**
25      * Construct a ShapeOnScreen.
26      *
27      * @param shape the Shape
28      * @param x its x coordinate
29      * @param y its y coordinate
30      */
31
32     public ShapeOnScreen( Shape shape, int x, int y )
33     {
34         this.shape = shape;
35         this.x     = x;
36         this.y     = y;
37     }
38
39     /**
40      * What Shape does this ShapeOnScreen represent?
41      *
42      * @return the Shape.
43      */
44
45     public Shape getShape() {
46         return shape;
47     }
48
49     /**
50      * The current x coordinate of this ShapeOnScreen.
51      *
52      * @return the x coordinate.
53      */
54
55     public int getX() {
56         return x;

```

```

57     }
58
59     /**
60      * The current y coordinate of this ShapeOnScreen.
61      *
62      * @return the y coordinate.
63      */
64
65     public int getY() {
66         return y;
67     }
68
69     /**
70      * Unit test.
71      */
72
73     public static void main( String[] args ) {
74         ShapeOnScreen sos = new ShapeOnScreen( null, 5, 7);
75         System.out.println("Shape: " + sos.getShape());
76         System.out.println("x: " + sos.getX());
77         System.out.println("y: " + sos.getY());
78     }
79 }

```